Presentation Attack Detection in Facial Biometric Authentication

A Project

Presented to

The Faculty of the Department of Computer Science

San José State University

In Partial Fulfillment

of the Requirements for the Degree

Master of Science

By

Hardik Kumar

May 2021

The Designated Project Committee Approves the Project Titled

Presentation Attack Detection in Facial Biometric Authentication

By

Hardik Kumar

APPROVED FOR THE DEPARTMENT OF COMPUTER SCIENCE

SAN JOSÉ STATE UNIVERSITY

May 2021

Dr. Fabio Di Troia	Department of Computer Science
Dr. Nada Attar	Department of Computer Science
Dr. Navrati Saxena	Department of Computer Science

ABSTRACT

Biometric systems are referred to those structures that enable recognizing an individual, or specifically a characteristic, using biometric data and mathematical algorithms. These are known to be widely employed in various organizations and companies, mostly as authentication systems. Biometric authentic systems are usually much more secure than a classic one, however they also have some loopholes. Presentation attacks indicate those attacks which spoof the biometric systems or sensors. The presentation attacks covered in this project are: photo attacks and deepfake attacks. In the case of photo attacks, it is observed that interactive action check like Eye Blinking proves efficient in detecting liveness. The Convolutional Neural Network (CNN) model trained on the dataset gave 95% accuracy. In the case of deepfake attacks, it is found out that the deepfake videos and photos are generated by complex Generative Adversarial Networks (GANs) and are difficult for human eye to figure out. However, through experiments, it was observed that comprehensive analysis on the frequency domain divulges a lot of vulnerabilities in the GAN generated images. This makes it easier to separate these fake face images from real live faces. The project documents that with frequency analysis, simple linear models as well as complex models give high accuracy results. The models are trained on StyleGAN generated fake images, Flickr-Faces-HQ Dataset and *Reface* app generated video dataset. Logistic Regression turns out to be the best classifier with test accuracies of 99.67% and 97.96% on two different datasets. Future research can be conducted on different types of presentation attacks like using video, 3-D rendered face mask or advanced GAN generated deepfakes.

***Keywords* – Biometric authentication, Convolutional Neural Network, Deepfake, Generative Adversarial Networks**

ACKNOWLEDGMENTS

I would like to thank my mentor Dr. Fabio Di Troia for his continued guidance and valuable suggestions in the project. He encouraged me throughout the course of the project to efficiently fulfill the proposed plan.

I would also like to thank my committee members Dr. Nada Attar and Dr. Navrati Saxena for taking interest in my project and providing constructive feedback.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

1. Introduction

Biometric Authentication is an innovative technology which recognizes and authenticates individuals based on some physical features like face landmarks, or retina or fingerprints, which are specific and unique to them. These features, called as ‘biometrics’ or ‘biometric data’ serve as identifiable and verifiable data streams which cannot be easily replicated. Hence, protection against presentation attacks which use photos or even deepfake technology is necessary for securing biometric systems.

The deployment of biometric systems worldwide has been motivated by the immense need of secure and definitive identification and authentication. A few application examples are as such :

- Over 2 billion electronic passports are valid today, which include the individual’s fingerprints and ICAO-compliant picture [1].
- The biometric identification scheme in the country of India integrates the demographic and biometric data of over a billion residents [1].
- Biometric authentication systems also play a crucial part in eKYC procedures, border control, military, immigration systems and many more [1].

Now, Spoofing or Presentation Attacks make the system vulnerable and reveals the possibility of loopholes in an otherwise novel and efficient system. Spoofing attacks include fabricated face masks or fingerprints, textured contact lenses, and even a recorded video of the authenticated individual, breaking the liveness detection module of the authentication pipeline. As the biometric systems are crucial wherever they are implemented, failure to prevent spoofing or presentation attacks results in serious consequences. This project mainly focusses on the face spoof attack prevention which focusses on, according to previous studies, texture and motion analysis as well as artificial analysis. In the past, a lot of different ways have been used to conduct face spoofing attacks like following:

- 2D static attacks are made with high-definition face pictures on flat paper masks.
- Fabricated lenses are used to bypass retina authentication systems.
- Impersonators even use wax or resin heads, 3D prints and sculptures.
- More recent 2D dynamic potential attacks include digital doubles or “deepfake” puppets or images.

This project focusses on handling two type of spoofing attacks: 2D attacks using photos or fabricated flat face picture, and deepfake photo and video attacks. In photo attacks, the impersonator shows a paper cutout or a captured photo of the authorized individual, to bypass the authentication system. Some weak systems just focus on detecting a face from the input feed, match the facial features with the features present in the database and authorize on a significant match. Although various machine learning techniques used in these systems have proven to be effective in a lot of cases, this approach is flawed, and spoofing techniques can easily bypass such systems using a face mask or displaying a photo in front of the webcam or input feed. In recent years, presentation attacks have evolved much more and become smarter to fool various authentication systems. To combat this, various techniques have been used in the past, like analyzing the Binary Statistical Features [2] of an Image , detecting focus and gaussian blurs from the input face, and also studying specific reflection patterns [1].

The other type of attacks is using “deep fake” technology, by injecting deep fake videos or photos in the input to fool the authentication system by impersonating as someone else. In the recent years, it has become significantly unchallenging to tamper a video by swapping a person's face with another person, and that too so effectively that it is difficult for the human to question its authenticity. Recent advances in video and audio editing, and especially Generative Adversarial Networks (GANs) allow for generation of synthetic "photographs" and videos of people who might have never existed, which are used for malicious purposes by an attacker. Various public scandals like celebrities' faces being swapped onto pornographic videos, fake politician speeches etc. necessitate the need to detect such deepfakes efficiently. It is not just confined to images or videos, but even to voice. In 2019, a U.K. based energy firm transferred €220,000 to a fake supplier in Hungary because a caller, posing as the leader of the firm's parent company by using deepfake technology to generate a fabricated voice, ordered the company's CEO to do so [3]. Such incidents have led to much more awareness in this area, and new gave rise to extensive research in the recent time. Various security systems are trained to detect minor distortions such as unexpected shadows, unnatural glare on glasses etc. which makes them detect any deepfake injections. A few research works have been done on degraded visual quality of images generated by GANs, which can be recognized by a trained deep learning model [4]. However, GAN-generated deepfake videos are still challenging for existing systems using artificial intelligence and machine learning technologies, and further development of GANs and face swapping technology makes it even more challenging.

In this project, the implementation and the work are carried out in two phases. The first phase handles the photo attacks by working on interactive action check of the person trying to be authenticated. In it, we are focusing on matching the face of the person in front of the webcam with an authorized individual's face in the biometric database and detect eye blinking to check for liveness and interactive check. In the second phase, we handle the deepfake attacks by training and testing various models to detect deepfake faces from real faces in an input image or video. This includes using various machine learning techniques to work on publicly available datasets of real faces and fake faces generated by GANs like StyleGAN [22] and also from apps like “Reface” [10] which use frameworks like DeepFaceLab [11], which is an open source deepfake face swapping system.

In Section 2, we put emphasis on various techniques and state-of-the-art machine learning and deep learning procedures used in the past to handle presentation attacks. Multiple research papers, journals and articles have been used as sources for background review and related work. In Section 3, we explain the methodology implemented in our project to handle the specific types of presentation attacks mentioned above. In Section 4, we go over the implementation of the methodology proposed and the experimental setup. In Section 5, we show the results and the observations of our experiments and lastly in Section 6, we present the conclusions of our project and the scope of future work.

2. Background and Related Work

In this section, we discuss about the background of presentation attack analysis, their types and various techniques used in the past to tackle them. Also, we explain the architectures and processes in detail as to how facial recognition works, and then presentation or liveness attacks work in different scenarios tampering the facial recognition systems. We also point out some of the drawbacks of the procedures used, and even challenges, which make it difficult to tackle the problem of 2D photo attacks and deepfake attacks, which continuously arise due to advancement in technology.

2.1 Facial Recognition

Face Recognition, or facial recognition, is one of the biggest areas of research within the field of computer vision. It is the process of using a face, precisely its features, to recognize or verify a person’s identity. Facial recognition is actually a 3-step process with each step performed in succession to get the outcome [1]:

- **Face Detection**: Location and Detection of human face(s) from an image or a video.
- **Face Capture**: Transformation of face or the facial features into a set of digital information like feature vectors.
- **Face Match**: Verification by matching the input face to a face present in the biometric database.

There are various applications of facial recognition in today’s systems. A useful example of it is of the recent US Capitol Riot incident. A Washington DC area student created a website called “Faces of the Riot” where he used an open-source facial extraction app to detect over 6,000 face images of Capitol Hill rioters from 827 videos posted on Parler social networking service [1].

FBI can investigate and identify the rioters by conducting the face capture and face matching process in the existing facial records or databases.

On further research, we explored certain deep learning techniques and architectures which play an important role in the process of facial recognition and even detecting presentation attacks. We went through the architectures of Convolutional Neural Network (CNN) and Histogram of Oriented Gradients (HoG) in detail, which have been applied in a lot of facial recognition and presentation attack prevention.

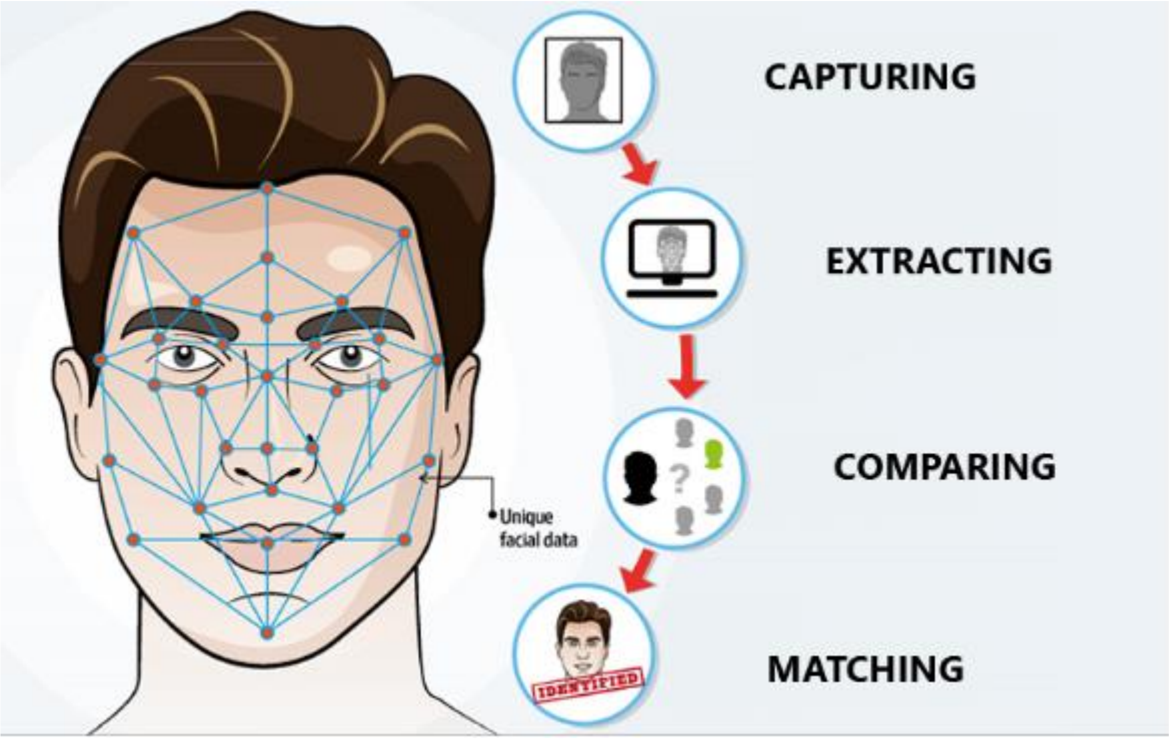


Fig.1 Facial Recognition pipeline [12]

2.1.1 Convolutional Neural Networks (CNN)

CNN, also called as ConvNets, came into existence during the 1990s but have received more recognition in the recent years due to their applications in Image Classification, Filtering and many more [5]. It is a deep learning technique which better works in classification of images in different classes by extracting useful information or significant features from an image and computing certain operations on them. Fig.2 shows a basic representation of a CNN architecture. The two main operations, namely Convolution and Pooling, are stacked together in combinations to form a complete CNN architecture.

The convolution layer is stacked before the pooling layer. The aim of the convolution layer is to extract high level significant features from an image [13] e.g. a face. Multiple feature maps, or the filter matrices, perform the convolution operation on the input image matrix and get the dot product, where each filter map might represent a distinguishing feature to be extracted like a nose, an eyebrow or an eye from a face. Fig.3 shows the convolution operation in a single layer.

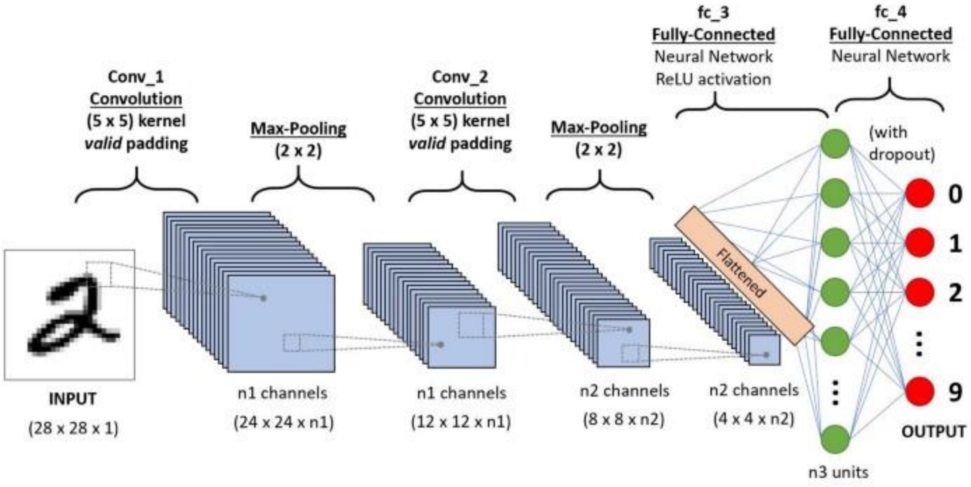


Fig.2 CNN architecture for Image Classification [13]

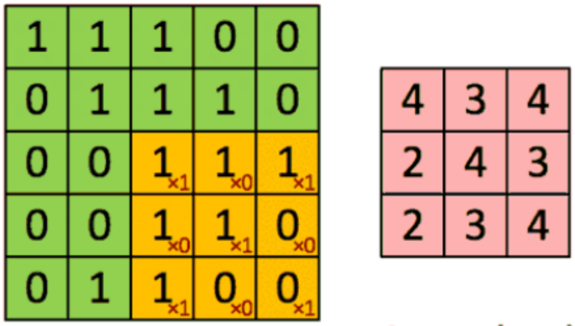


Fig.3 Convolution operation with a) image matrix and b) convolved feature (L-R) [13]

In addition, a pooling layer is generally stacked after a convolution layer. This is to highlight and add more weight to the more significant features extracted from the feature maps of the convolution layers [13]. There are various pooling types as well, like Max Pooling or Average Pooling, which are applied according to the use case and outputs [6]. The pooling layer also decreases the computational power need by removing the spatial size of the convolved features and introducing dimensionality reduction [13]. Fig.4 displays a pooling operation in a Max Pooling as well as Average Pooling layer.

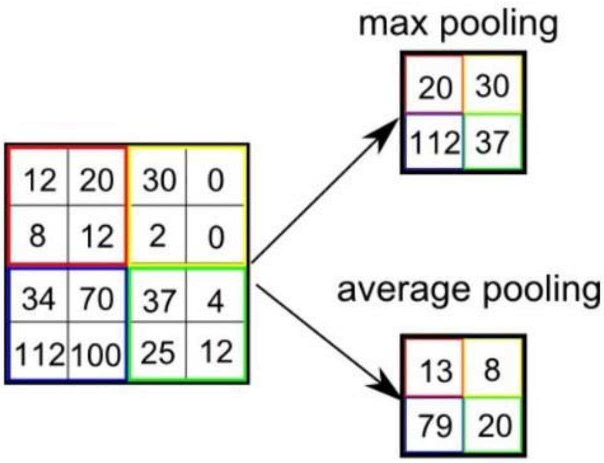


Fig.4 Image/convolved feature matrix, with max pooling and average pooling output [13]

2.1.2 Histogram of Oriented Gradients (HOG)

A HOG is a feature descriptor used in image processing and computer vision, generally for the purpose of object detection. They are widely known to be effective in the applications of pedestrian detection. It depends on the objects within an image having the property to have the intensity gradients' or edge directions' distribution [7]. It works on the idea of edge directions or local intensity gradients to characterize the appearance and shape of a local object. Histograms provide translational invariance and is also simple and fast to calculate, that is why it is an effective technique for object detection in real-time as well [8]. The HOG feature abridges the distribution of measurements within the regions of a given image and 1-D histograms of gradient directions or edge directions for all the key points of an image are assimilated and joined to make the final histogram feature. It proves to be useful for recognizing objects with textures or different shapes like a face with obstructed lighting as well. Fig.5 shows the whole process of HOG.

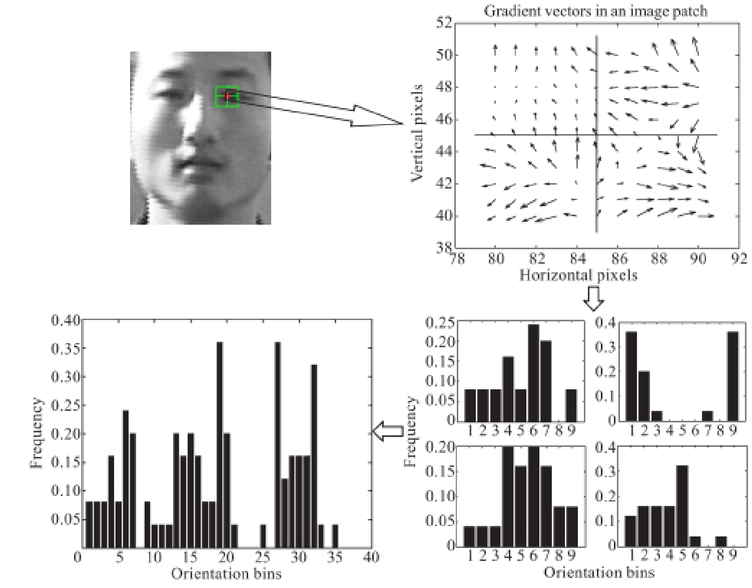


Fig.5 A typical HOG processing on a face image [8]

2.2 Presentation Attacks

Face Recognition, as known, is a widely used biometric approach and is implemented currently in various systems and companies. However, such systems are also quite vulnerable to spoofing attacks called presentation attacks. There are multiple ways to spoof such systems like using a portrait photograph, 3D rendered masks or deepfakes [9]. A lot of the face recognition systems are unsuccessful in distinguishing between a ‘live’ and a ‘fake’ face, which is the essence of “Liveness Detection”. Secure biometric systems consisting of face recognition also need to incorporate face liveness detection in order to provide security against such presentation attacks. Liveness detection is a very active research topic in case of fingerprint and iris recognition, but the approaches for this in face recognition have not been as extensive. However, various approaches have been researched into in recent years giving promising results. A few of the approaches use the texture and frequency analysis of the images, interactive action check and even the use of specific hardwares [14] especially for liveness detection in face recognition systems.

2.2.1 Texture and Frequency-Based Analysis

Kim et al. [2] use this approach in implementing face liveness detection system. The primary objective of the research is to distinguish between live faces and 2-D paper masks generally used by impersonators. In broader terms, they are distinguished on the basis of their image details and even the shape. For the feature-extraction phase, which is most important to detect liveness, the approach use three implementations: frequency-based, texture-based and fusion-based.

In frequency-based feature extraction, the authors have implemented power spectrum-based method, which utilizes the low and the high frequency information in the image to be used for classification. The frequency analysis is carried out because of two reasons. Firstly, the difference in the low frequency regions of a live 3-D image and a 2-D picture or paper mask is significant due to the illumination differences. Secondly, there is also a difference in high frequency information of images between the masks and live faces because of the difference in detail. The frequency analysis is performed using the 2-D discrete Fourier Transform.

In addition, Local Binary Pattern (LBP) method [2], which is dependent on description details, has been implemented for the texture-based feature extraction. LBP is a widely popular technique to be used for texture analysis of images. It computes a local representation of texture by comparing each pixel with the neighborhood pixels surrounding it. It checks whether the central pixel value is greater than or lesser than each of the surrounding pixel values and gives a binary value 1 or 0 for either case. Then, the 1-D LBP value array is converted to decimal, and this process is repeated for each pixel to obtain an LBP feature vector for the image.

For fusion-based feature extraction, the authors used Support Vector Machine (SVM) classifier for performing liveness detection with the feature vectors generated by power spectrum-based and LBP-based methods.

Experimental results showed that the fusion-based method was the most promising with an error rate of 4.42% compared to texture-based method and frequency-based method with error rates of 12.46% and 5.43% respectively.

2.2.2 Analysis Based on Variable Focusing

Kim et al. [15] implemented face liveness detection using variable focusing. The approach taken is to make use of the variation of pixel values by putting focus between two images taken in a sequence in different focuses. They emphasize on collecting the differences in focus values between fake and real face images when two sequential images, in focus and out focus, collected from each subject. It is observed that in fake faces or face photographs and printouts, there is very little difference in images taken from different focuses. However, in the case of real faces, the focuses regions in the image are clear while the unfocused regions are blur due to depth information. This analysis was on the basis of the degree of Depth of Field (DoF) and it turns out to be effective in distinguishing fake faces from real ones. DoF is defined by the range between the farthest and nearest points in a given image or object, currently in focus. For calculating the degree of focusing in images, the authors use Sum Modified Laplacian (SML) which represent the degree values as a second order differential filter which is transformed.

The closest and farthest points chosen for real 3-D faces and fake faces (printouts) are nose and ears, where first the nose (the nearest point) is in focus and in the next shot, the ears (the farthest point). Then, the SMLs of both the pictures are calculated, for each real and prinout face, and then the difference of SMLs is computed between the two differently focused pictures. Then, the sum difference of SMLs is calculated for each column for the one-dimensional analysis. Then, it was observed that the SML sum difference showed similar patterns when a real face was taken into focus, whereas the patterns of fake faces were inconsistent. This proved a distinctive ridge between real and fake faces. The experimental results showed that the False Acceptance Rate (FAR) was 2.86% and False Rejection Rate was 0% when DoF is small, which proved to be more promising than when DoF is large.

2.3 Deepfake

The second part of this project focuses on Deepfake detection, distinguishing between Deepfake and real images. Recently, there have been occurrences of a lot of fake face images and videos generated by digital manipulation and face swapping in particular with Deepfake methods [16]. This has become a big public concern due to the unrest and rumors it creates. “Deepfake” is a term, originated in 2017 by a reddit user “deepfakes”, referring to a deep learning technique able to create fake videos by either generating a completely different face which is not a real person, or by swapping the face of one person with another. Deepfakes have also recently been observed to be used for recreational and benign purposes like some face apps or filter apps. However, there have been multiple cases of the technology being used for harmful purposes, like creating fake pornography, fake news, financial frauds and many more. The videos and images generated by strong Deepfake software or technologies are quite deceiving and very difficult for human eye to detect it being not authentic [17]. Deepfake technology utilize powerful techniques of machine learning and artificial intelligence to generate such deceiving results. The main methods and architectures used for the generated of deepfakes are based on deep learning involved in training autoencoders, and mostly Generative Adversarial Networks (GANs).

2.3.1 Generative Adversarial Networks (GAN)

Generative Adversarial Networks, or GANs, are a deep learning-based generative model [18]. These are architectures or network to develop and train generative models. One of the most popular applications are generating examples for image datasets, image to image classification, generating realistic photographs from a fixed domain, and many more [19]. Ian J. Goodfellow et al. [20] first introduced the concept of these networks in their research paper in 2014. A GAN model is trained using two types of neural network models: the "generator" and the "discriminator". These two models work as an *adversary* to each other, hence the name. They are pitted against each other in a game theory sense competition, where the generator model tries to fool the discriminator model by generating fake samples and getting the discriminator to incorrectly classify them as real, whereas the discriminator works to fail the generator by trying to distinguish between real and fake generated images with maximum possible accuracy. Both the neural networks ultimately work on the same objective, to generate fabricated images or samples (in our case, face images) that correspond to a probability distribution p_g which is comparable to p_t , the probability of original data or domain.

Table 1: GAN Notation

Symbol	Description
G	Generator
D	Discriminator
z	Random noise belonging to probability distribution p_z
p_t	Probability distribution of true data samples
p_g	Probability distribution of G's output $G(z)$
$D(.)$	Output of D for any input
$G(.)$	Output of G for any input
$V(D, G)$	Cost function for the GAN

2.3.1.1 GAN Architecture and Working

The main symbols used in the mathematical notations of the working of GAN are mentioned in Table 1. As explained in the previous section, the generator *G* is trained in a way that the discriminator *D* classifies a fake face sample, *G(z),* as real. This is obtained by minimizing the expected value of log likelihood of **1 – D(G(z))** as shown in equation 1 below.

$$\min_{G} \mathbb{E}_{z \sim p_z}[\log 1 - D(G(z))]$$

On the other side, the discriminator *D* is trained in a way that on seeing a fake face sample $z \sim p_g$, it gives the probability almost equal to 0 to deem it as fake. Also, the discriminator is trained to classify real face sample, $x \sim p_t$, as real. This is done by maximizing the expected log likelihood value of **1 – D(G(z))** as shown in equation 2 below.

$$\max_{D} \mathbb{E}_{x \sim p_x}[\log D(x)] + \mathbb{E}_{z \sim p_z}[\log 1 - D(G(z))]$$

Now, equations 1 and 2 can be combined as the Generator’s functioning is independent of the discriminator’s predictions on true data. Hence, this gives a combined function ***V (D, G)*** as follows:

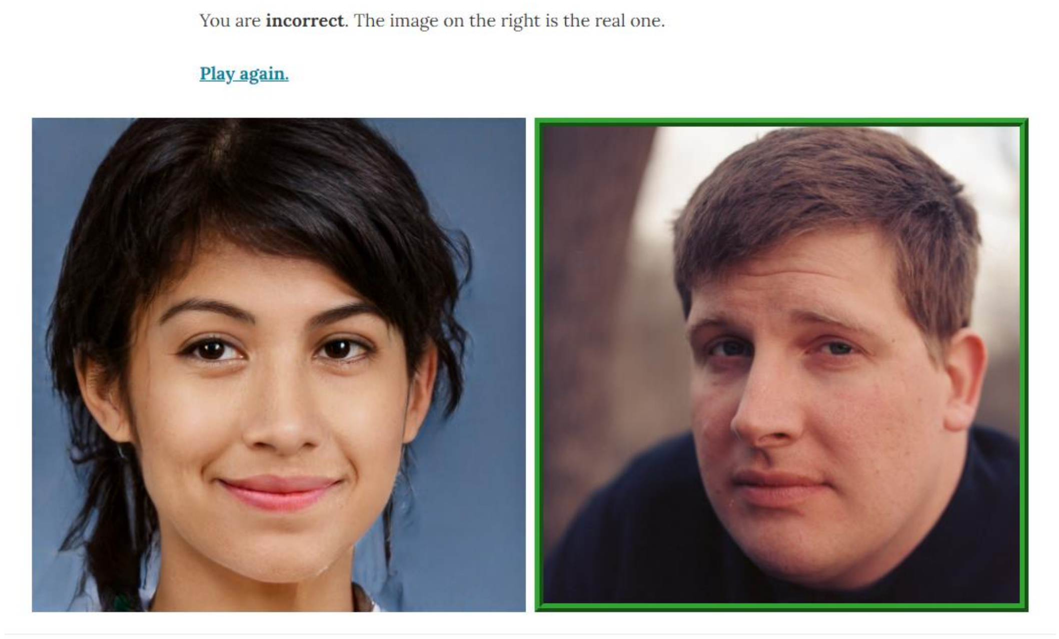
$$\min_{G} \max_{D} V(D, G) = \mathbb{E}_{x \sim p_x}[\log D(x)] + \mathbb{E}_{z \sim p_z}[\log 1 - D(G(z))]$$

This equation depicts the min-max game between the Generator and the Discriminator [20][21], to generate face samples strong enough to fool a lot of face recognition systems with weak classifiers or discriminators.

2.3.2 StyleGAN

Karras et al. [22] proposed an alternative generator architecture for GANs called "StyleGAN". In this approach, it inherits techniques and ideas from style transfer literature. The authors highlight a few drawbacks and weaknesses in the other GAN architectures. One of them is that usual GAN models lack a deep and extensive understanding of the origin of stochastic gradients, which is one of the key characteristics of image synthesis. In addition, lack of understanding in the latent space and the lack of quantitative ways to compare different generators is also shown.

The authors propose that in StyleGAN, they have redesigned the generator architecture to control image synthesis in a much better way. The generator adjusts the "style" of images at each convolution layer based on the latent code. This helps it control the strength of image features at different scales. Also, this new architecture leads to an automatic, unsupervised dissociation of high-level attributes, like pose and identity of human faces, from stochastic variations, like hair texture, freckles etc. This makes the image synthesis to be scale-specific and intuitive, which gives much more promising results than a classic GAN architecture. Even though the modified architecture makes changes in the generator structure, there is no change in the discriminators or the loss function. Hence, the discriminator and the loss function are just like in any other GAN model. The authors have publicly released the dataset consisting of images generated by StyleGAN, with the true data being images from the Flickr-Faces-HQ dataset (FFHQ) [30]. The fake face images generated by StyleGAN are much more deceiving to the human eye in comparison, as demonstrated at *whichfaceisreal.com* [23], where human accuracy peaked at 75% in guessing whether the face is real or fake [23]. Fig.6 shows an example of this.



You are **incorrect**. The image on the right is the real one.

Play again.

Fig.6 (L-R) Deepfake image generated by StyleGAN; an image from FFHQ Dataset [23]

2.3.3 Vulnerability Assessment and Frequency Analysis for Deepfake Image Recognition

Korshunov and Marcel [4] worked on distinguishing deepfakes from real videos by analyzing various aspects of the videos. They work on high-quality and low-quality versions of deepfake videos for each subject with face swapping generated by GANs. First, they used an audio-visual approach in which it analyzes and detects incompatibilities between the audios and the lip movements in the deepfake videos. This approach focused on analyzing the efficiency of Deepfake to copy mouth and lip movements with face swapping. Second, they also applied various baseline methods to analyze deepfake videos, to target multiple aspects and find out different kind of vulnerabilities in Deepfake videos. The approaches and methods used are Linear Discriminant Analysis (LDA), study on Image Quality Metrics using Support Vector Machine (SVM) classifier and Principal Quality Analysis (PCA). In the results, it was observed that state-of-the-art face recognition algorithms based on Facenet and VGG were ineffective in protection against deepfake videos, with an error rate of 95%. The best approach turned out to be the use of techniques based on Image Quality Metrics with SVM classifier, giving only 8.97% error rate with high quality deepfake videos.

The other approach is given by Frank et. Al [24] where the deep fake images are analyzed in the frequency domain rather than the image domain. This is an approach generally used in the field of image forensics. The main focus in the research paper is on the following things:

- Extensive frequency domain analysis of the images which belong to the real face data set and also the images generated by popular GAN models. This study in the frequency domain showed the presence of some critical features in the GAN generated images differentiating them from the real images.
- On further investigation, it was found out that the features or artifacts are born due to the upsampling operations used by GANs to generate the fake face images. This highlights a vulnerability of GAN models in mapping from a low-dimension latent space to a higher one.
- Ultimately, the authors perform comprehensive comparisons between the frequency domain approach and classical state-of-the-art approaches. The comparisons prove the former to be much more effective against deepfake presentation attacks in spite of using fewer parameters.

For the frequency domain analysis, the images are transformed using Discrete Cosine Transform (DCT). DCT indicates a finite sequence of data points, in an image or a signal, in terms of a cosine functions’ sum oscillating at different frequencies. It was first proposed in 1972 and is still widely used as a transformation technique for image compression and signal processing [24]. The log-scale of DCT coefficients is used for the analysis, due to the varying range of coefficient magnitude values at low frequency and high frequency. This is used because the corresponding spatial frequency contribution to the image is depicted by the magnitude of each coefficient. It is observed that in frequency spectrum real natural face images, most pixels are correlated to each other and change slowly and not abruptly, hence low-frequency functions are used to approximate large sections of the image. On the other hand, images generated by GANs have several high frequency components and high coefficient magnitudes throughout the frequency spectrum. These artifacts are theoretically attributed to the various upsampling operations like Nearest Neighbor, Bilinear etc. used in GAN image synthesis.

This particular approach is implemented in our project, and this transformation of type-II DCT is used to be fed into multiple classifiers to see the accuracy results on different dataset of images generated by different GANs.

3. Methodology

In this section, we detail the pipeline for the implementation of our project for both the parts: photo presentation attack and deepfake detection. This section explains the idea and the techniques used in detail.

3.1 Photo Attack Presentation Attack Detection

This is to detect face liveness by interaction check, which is eye blinking. The methodology adopted for this phase is explained below:

1. Install the *face_recognition* library in Python [25]. This is built using C++ toolkit called Dlib [26] containing machine learning algorithms to solve real world problems. This library is used to save the face encodings of the authorized or known face, by first detecting the face using *face_locations* method and then encoding the face into a 128-length vector using *face_encodings* method and save it. We used my face to be used as the saved face and saved the encodings.
2. The *face_locations* method can find face locations in an image using two methods: HOG or CNN. As explained in the background section, HOG is an effective feature descriptor. Also, as HOG is much faster than CNN and the face locations need to be done in real time for the webcam feed, HOG is chosen over CNN.
3. Save the encodings for the known face or authorized face using *face_encodings* method of the *face_recognition* library. The encodings would be saved as a 128-length feature vector.

4. For face liveness detection, train a model for classifying whether the eyes are closed or open in a face. The aim is to detect blinking i.e. open-closed-open eye pattern . In this project, we train a CNN model for classification.
5. Now, to detect the faces in the webcam in real time, use OpenCV's pretrained Haar-cascade classifier [27]. It is a machine learning technique where a significant amount of positive and negative images trains a cascade function. It is widely used for face detection, eye detection and object detection in the field of computer vision.
6. Now, after detecting face and eyes in each frame of the webcam input, compute the face encodings of the input face and match them with the saved encodings. If it is the same, display the name of the individual.
7. After the face is recognized, the detection of eye blinking is required for liveness detection. This is the most important process for detecting whether the input face is a live face or a paper mask or a picture. If the open-closed-open pattern is detected, the person is authorized and is classified as 'live'.

3.2 Deepfake Detection

This is to detect face liveness by separating deepfake face images, generated by GAN models, from real face images. The methodology is as follows:

1. Make two datasets with one having real images from FFHQ Dataset and fake images generated by StyleGAN trained on FFHQ dataset. The other dataset has real images from the same FFHQ Dataset. However, the fake images are generated from the mobile app *Reface* [10]. This app generates deepfake videos for recreational purposes by swapping the user's face on top of a video. The app generates deepfake videos through a

combination of GANs and DeepFaceLab [11] software, which also inherently uses GANs.

2. Then, compute Type-2 2D-DCT of the images. This intuition is inspired from [24] where the images are analyzed in the frequency domain. As explained in the background, DCT is most widely used for frequency analysis in the field of image processing, and also for image compaction. Hence, due to such factors, applying DCT enables analyzing the images in frequency domain without focusing on much parameters. The type-2 2D-DCT is calculated as follows:

   Let input image be the matrix $I \in R^{N_1\ x\ N_2}$, where the pixel values are given by $I_{x,y}$. The 2D-DCT is expressed as function $D: \ R^{N_1\ x\ N_2} \rightarrow R^{N_1\ x\ N_2}$. This functions maps an image $I = \{ I_{x,y} \}$ to its frequency representation $D = \{ D_{kx,ky} \}$ which is given as :

$$D_{k_x,k_y} = w(k_x)w(k_y)\sum_{x=0}^{N_1-1}\sum_{y=0}^{N_2-1} I_{x,y} \cos\left[\frac{\pi}{N_1}\left(x+\frac{1}{2}\right)k_x\right]\cos\left[\frac{\pi}{N_2}\left(y+\frac{1}{2}\right)k_y\right]$$

   for all $k_x = 0, 1, 2, \dots, N_1 - 1$ and for all $k_y = 0, 1, 2, \dots, N_2 - 1$

   and $\omega(0) = \sqrt{1/4N}$ and $\omega(\mathrm{k}) = \sqrt{1/2N}$ for k > 0

3. Compute the log of the DCT coefficients to handle large magnitude values of high frequency regions in images. Also, normalize the dataset.

4. Then, after the dataset is preprocessed, split the dataset into train, validation and test set in the ratio ~ 70:15:15.

5. Then, train multiple machine learning and deep learning classifiers on the dataset. The classifiers used are CNN, Resnet, Multinomial Logistic Regression and SVM.

4. Implementation

In this section, we give a detailed explanation of the dataset used for both the photo attack phase and deepfake detection phase of the project, the models and the machine learning and deep learning techniques used to analyze the results.

4.1 Photo Attack Presentation Attack Detection

In this phase of the project, as explained in the previous section, we analyze the eye blinking detection of the face shown in front of the webcam. The procedure is to first save the face encodings of the authorized individual, detect the face and eyes in each frame of the webcam feed, match the face encodings of the webcam feed face with the saved authorized face's encodings, and then detect the eye blinking.

4.1.1 Dataset

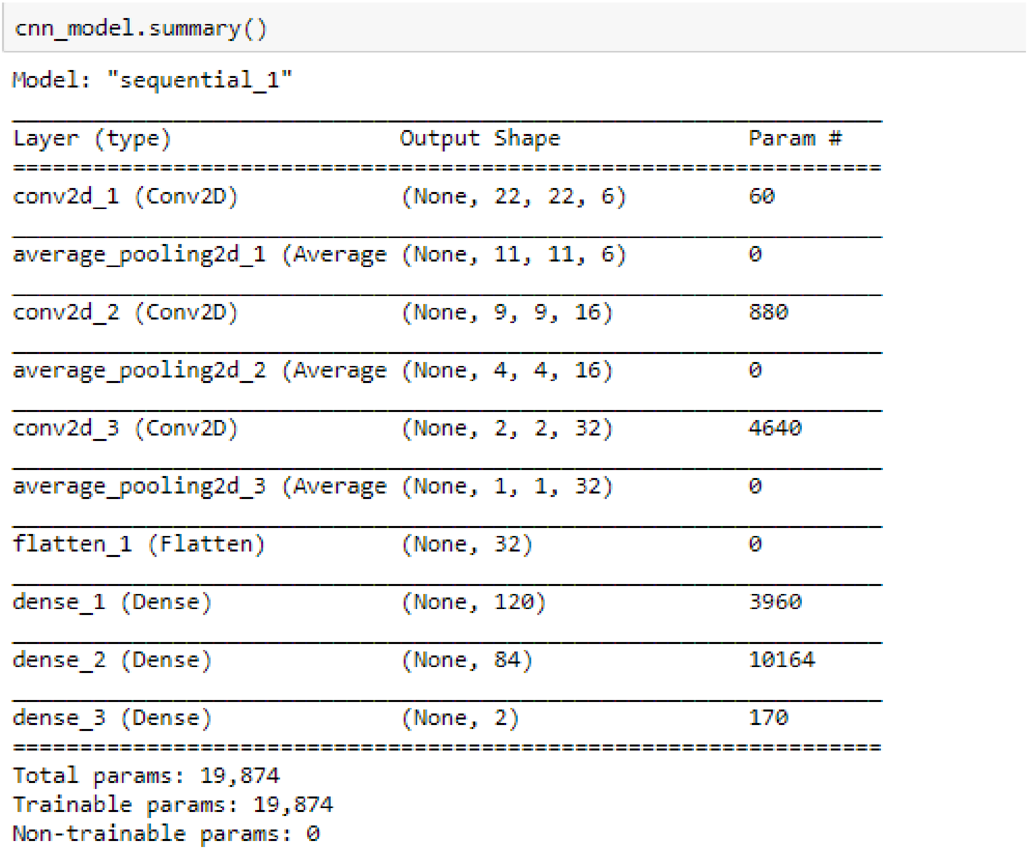
For this phase, we used the Closed Eyes in the Wild (CEW) dataset [28]. This dataset contains sets of Facial images in full resolution, Facial images in compressed size of 100 X 100, and set of eye images in size 24 X 24. We chose the eye images set. This set contains 2423 subjects with 1192 subjects with both eyes closed and are scraped from the internet, and the other 1231 subjects with both eyes open are collected from Labeled Face in the Wild [29] dataset. The dataset contains a total of 4848 images of open and closed eyes in the dataset. The split between the train, validation and test data is approximately in the 70:15:15 ratio. This dataset is used to detect closed eyes and open eyes in the face shown in front of the webcam.

4.1.2 Phase I Implementation

- First, we installed the *face_recognition* library from Python. Used the Histogram of Gradients (HOG) train method to find face locations and encodings
- Second, used a CNN based model to train on the CEW dataset for the prediction of open and closed eye of the face shown. The architecture of the model is as follows:
    - Convolutional Layer with 6 convolutional filters/feature maps and ‘relu’ activation
    - Subsampling Layer with Average Pooling
    - Convolutional Layer with 16 convolutional filters/feature maps and ‘relu’ activation
    - Subsampling Layer with Average Pooling
    - Convolutional Layer with 32 convolutional filters/feature maps and ‘relu’ activation
    - Subsampling Layer with Average Pooling
    - Flatten layer to reshape the output of convolutional layers into a 1-D feature vector
    - Fully Connected Layer (Dense) with 120 neurons and ‘relu’ activation
    - Fully Connected Layer (Dense) with 84 neurons and ‘relu’ activation
    - Fully Connected Layer (Dense) with 2 neurons and ‘softmax’ activation, for output
    - Categorical CrossEntropy Loss function used and Adam optimizer used

  Fig.7 shows the model summary.
- Ran the python program to take the video input from the webcam, used OpenCV’s “Haar-cascade classifiers” to detect the face and eyes in each frame of the webcam input. Then, with “compare_faces” method of face_recognition library, the encodings of each

face are matched to the saved face encodings. And, the eyes are classified as 0 or 1 by the classifier trained in the previous step. The classifications for each frame are stored as a 1-D binary vector. Now, if the authorized person blinks in front of the webcam, the model will detect that the eyes were open, then closed, then open. So, a pattern of ‘01’ or ‘10’ will be encountered, which would mean that the blinking is detected and would deem the person infront of the webcam as ‘live’.



```
cnn_model.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
conv2d_1 (Conv2D)            (None, 22, 22, 6)         60        
_________________________________________________________________
average_pooling2d_1 (Average (None, 11, 11, 6)         0         
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 9, 9, 16)          880       
_________________________________________________________________
average_pooling2d_2 (Average (None, 4, 4, 16)          0         
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 2, 2, 32)          4640      
_________________________________________________________________
average_pooling2d_3 (Average (None, 1, 1, 32)          0         
_________________________________________________________________
flatten_1 (Flatten)          (None, 32)                0         
_________________________________________________________________
dense_1 (Dense)              (None, 120)               3960      
_________________________________________________________________
dense_2 (Dense)              (None, 84)                10164     
_________________________________________________________________
dense_3 (Dense)              (None, 2)                 170       
=================================================================
Total params: 19,874
Trainable params: 19,874
Non-trainable params: 0
```

Fig.7 Model summary for Eye Detection

4.2 Deepfake Detection

For this part of the project, we focused on face liveness detection by ensuring protection against deepfakes. We use several classifiers and techniques to separate deepfake face images from real face images. Also, further analysis is performed on deepfake images generated by GANs to find out vulnerabilities and artifacts.

4.2.1 Dataset

We have used 2 set of datasets for the classification of deepfakes. The two sets are as follows:

- **Set 1**: The dataset includes real face images from the FFHQ [30] and the fake images are generated by StyleGAN, which is pre-trained on FFHQ dataset. The split of the dataset is shown in Table 2.
- **Set 2**: The dataset includes real face images from the same FFHQ dataset. The fake images are generated from the mobile app *Reface*. The split of the dataset is shown in Table 3.

Table 2: Set-1 Dataset split

Classes	**Train Images**	**Validation Images**	**Test Images**	**Total Images**
Fake (StyleGAN)	23800	5100	5100	34000
Real (FFHQ)	35000	7500	7500	50000

Table 3: Set-2 Dataset split

Classes	Train Images	Validation Images	Test Images	Total Images
Fake (Reface)	9754	2090	2090	13934
Real (FFHQ)	14000	3000	3000	20000

4.2.2 Phase II Implementation

Firstly, data preprocessing is performed to make the dataset ready for classification. For the data preprocessing, the Type-2 DCT coefficients are computed for each image. Type-2 DCT coefficients are computed using *fftpack* module of *scipy* [31] library in Python. Then, the log of those coefficients is computed to handle high magnitudes of DCT coefficients. In addition to that, the resulting dataset is then normalized. The resulting dataset is saved as tfrecords file.

After the data is preprocessed and normalized, multiple classifiers are trained on the dataset. As there are 2 sets of datasets made, the classifiers are trained on each set and provide the result. The following classifiers are used:

- **CNN**: The architecture of CNN used is as follows :
  - Convolution Layer with 6 filters and 'relu' activation
  - Convolution Layer with 8 filters and 'relu' activation
  - Average Pooling 2D Layer
  - Convolution Layer with 16 filters and 'relu' activation
  - Average Pooling 2D Layer
  - Convolution Layer with 32 filters and 'relu' activation
  - Flatten Layer to convert into 1-D feature vector
  - Fully Connected Dense Layer with 'sigmoid' activation

    - Binary_crossentropy used as Loss Function, and Adam as optimizer

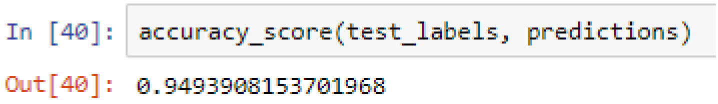
- Multinomial Regression Classifiers are used with varying kernel regularizer values
- ResNet Architecture
- Support Vector Machine (SVM)

5. Results and Discussion

In this section, we detail the results of the experiments performed in our project. The results are mentioned separately for the two parts of the project.

5.1 Photo Attack Presentation Attack Detection

In this, firstly the model is trained on the CEW dataset which consists of open and closed eyes. The eye detection model is trained to detect open eyes and closed eyes whenever a face is detected in thr webcam. The CNN-model is trained for 20 epochs, giving the accuracy of **94.93%** as shown in Fig.8.



```
In [40]: accuracy_score(test_labels, predictions)
Out[40]: 0.9493908153701968
```

Fig.8 Accuracy Score for Eye Detection Model

The accuracy and loss graphs for the model training are also shown in Fig.9 and Fig.10 respectively.

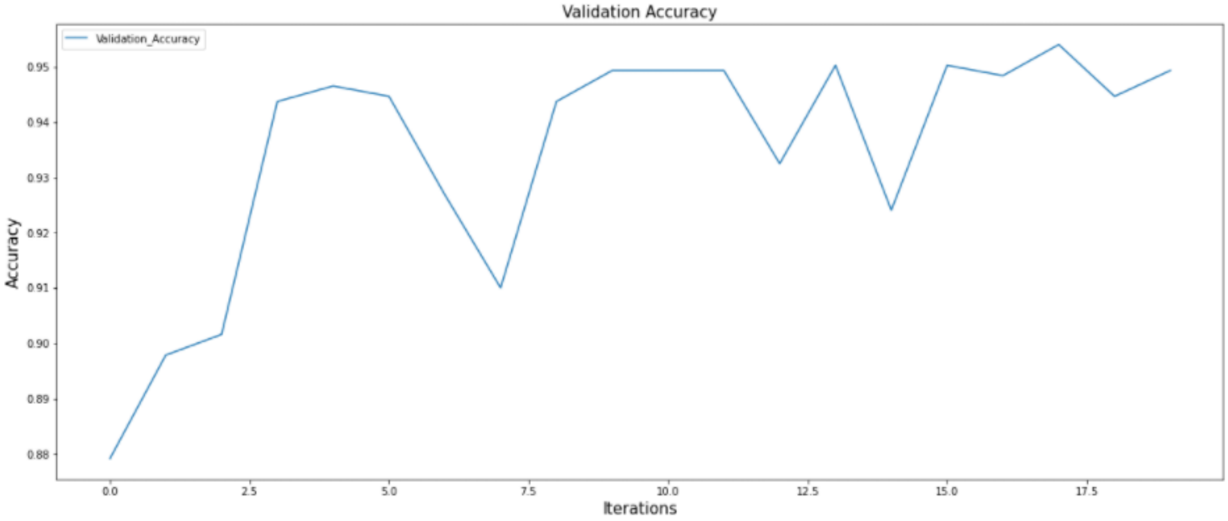


Fig.9 Accuracy Graph for Eye Detection Model

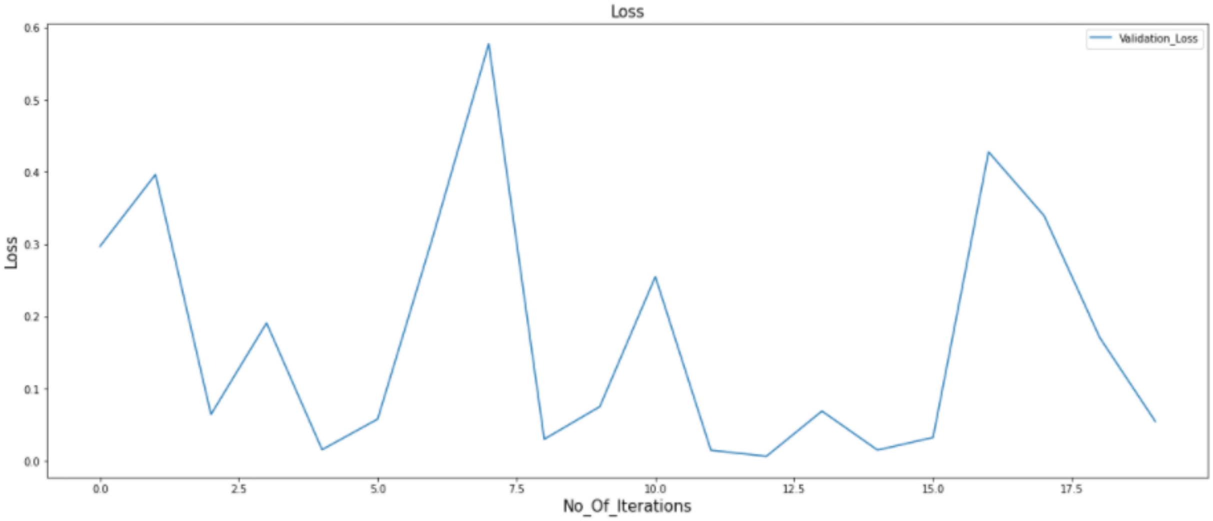


Fig.10 Loss Graph for Eye Detection Model

It can be observed in the accuracy graph, the trend of the learning curve is increasing with each epoch. It shows that the model is learning efficiently and not overfitting. Also, in the loss graph, the variation in slope is much more in comparison with a few spikes. However, still the overall trend is decreasing.

The confusion matrix is also shown in Fig.11 representing the values of correctly classified ‘eyeClosed’ and ‘eyeOpen’ and incorrectly classified as well.

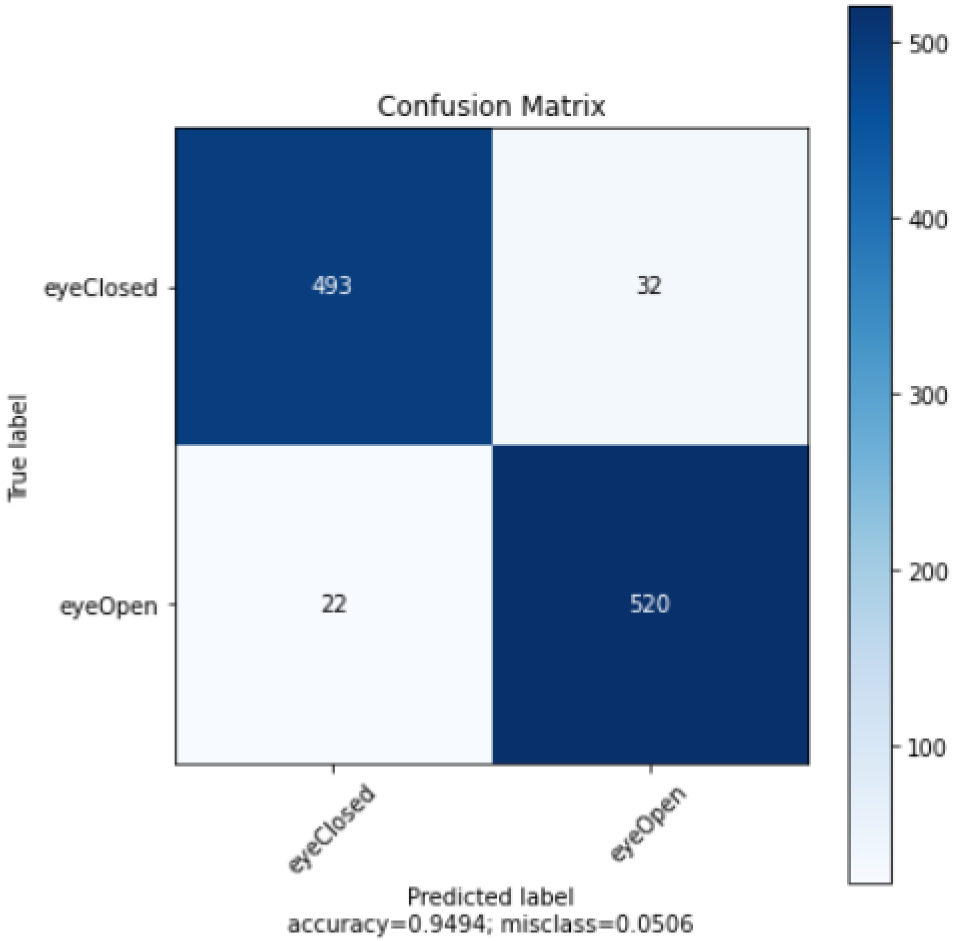
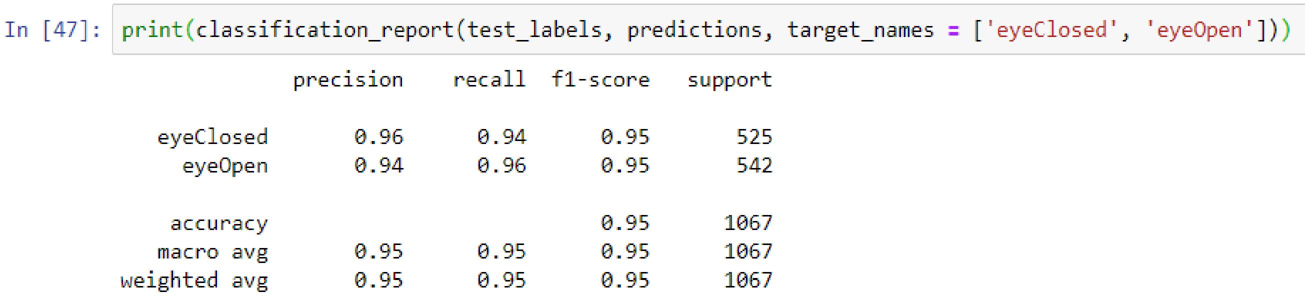


Fig.11 Confusion matrix for Eye Detection Model

In the confusion matrix, the row indices are the actual labels of the images, and the column indices are the predicted values. It can be seen that:

- Correctly predicted closed Eyes (True Positive): 493
- Closed Eyes predicted as Open (False Negative): 32
- Correctly predicted Open Eyes (True Negative): 520
- Open Eyes predicted as Closed (False Positive): 22

Hence, the precision and recall values for each class are shown in the classification report in Fig.12.



```
In [47]: print(classification_report(test_labels, predictions, target_names = ['eyeClosed', 'eyeOpen']))

              precision    recall  f1-score   support

   eyeClosed       0.96      0.94      0.95       525
     eyeOpen       0.94      0.96      0.95       542

    accuracy                           0.95      1067
   macro avg       0.95      0.95      0.95      1067
weighted avg       0.95      0.95      0.95      1067
```

Fig.12 Classification report for Eye Detection Model

After the model is trained, this model is used to detect the Eye Blinking pattern on each face of each frame of the webcam video input. The eye blinking is detected only for the authorized individuals. Fig.13 and Fig.14 shows an example of the demo when the blinking is not detected, and when it is.

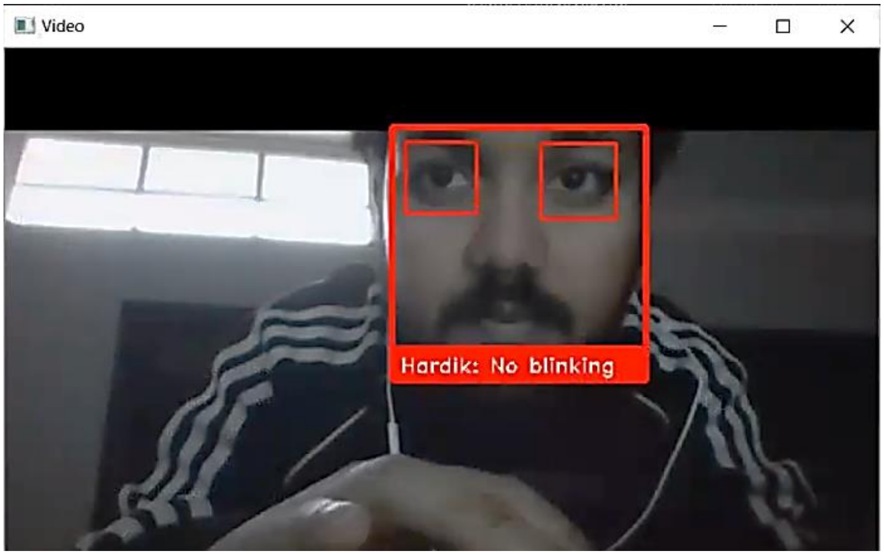


Fig.13 Webcam feed recognizing the face, but no blinking detected

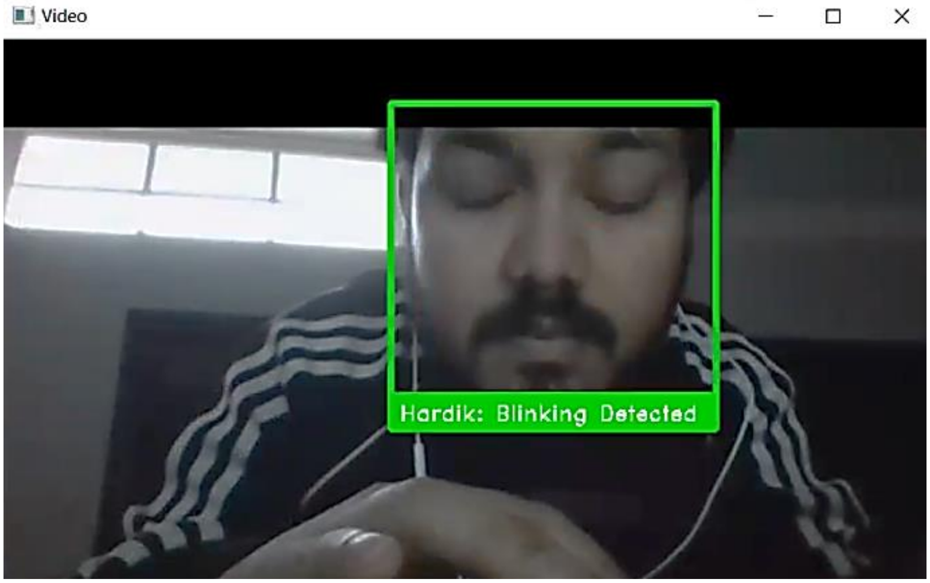


Fig.14 The model recognizing that blinking is detected

5.2 Deepfake Detection

In this phase, multiple models or classifiers have been trained on the two different sets of the datasets mentioned in the methodology section. It is observed that the CNN-based classifiers and the linear classifiers performed with high accuracies on the datasets. This gives the inference that the Type-2 DCT of images makes it easier to distinguish between real face images and GAN generated images. The following results also support and confirm the results presented in [24].

5.2.1 Set-1 Results

In this set, the classes consist of Real Face Images and the images generated by StyleGAN [22]. The models trained are CNN, ResNet architecture, SVM and Logistic Regression.

5.2.1.1 CNN

Figures 15 and 16 show the accuracy and loss graphs for CNN after training on Set-1.

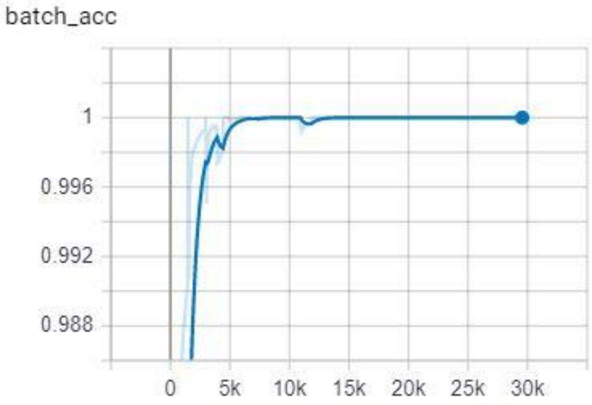


Fig.15 CNN Batch Accuracy vs Steps (Set-1)

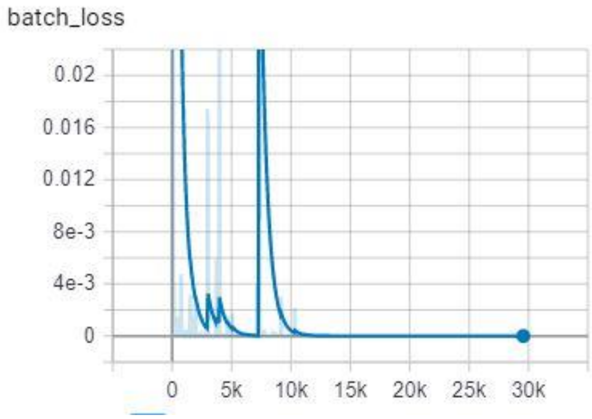


Fig.16 CNN Batch Loss vs Steps (Set-1)

The test accuracy for the CNN model turned out to be **100%** proving it to be the most effective for Set-1.

5.2.1.2 ResNet

Figures 17 and 18 show the accuracy and loss graphs for ResNet after training on Set-1.

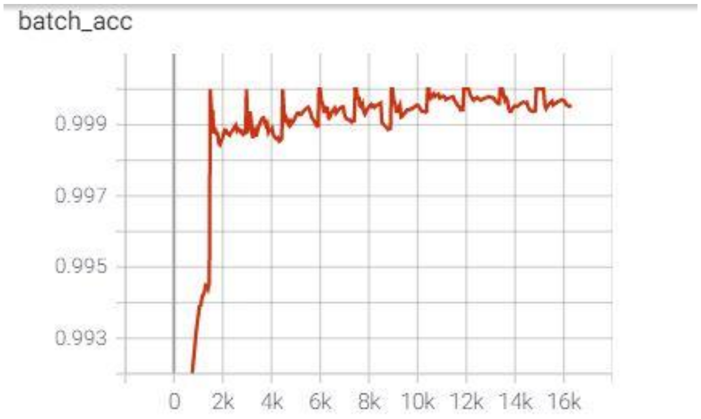


Fig.17 ResNet Batch Accuracy vs Steps (Set-1)

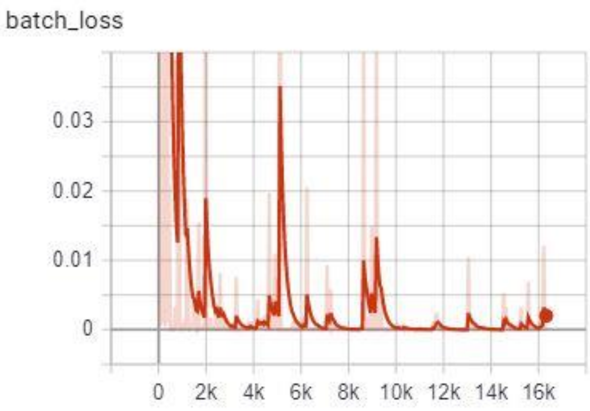


Fig.18 ResNet Batch Loss vs Steps (Set-1)

ResNet gave the test accuracy peak of **99.94%.**

5.2.1.3 SVM

Figures 19 and 20 show the accuracy and loss graphs for SVM after training on Set-1.

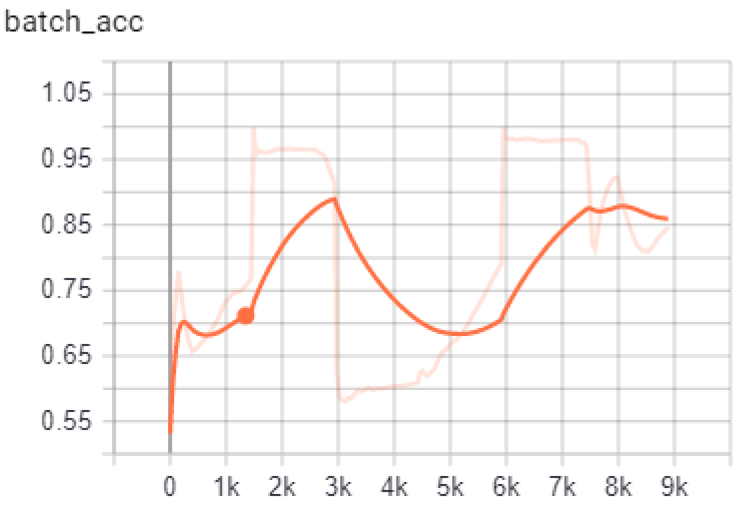


Fig.19 SVM Batch Accuracy vs Steps (Set-1)

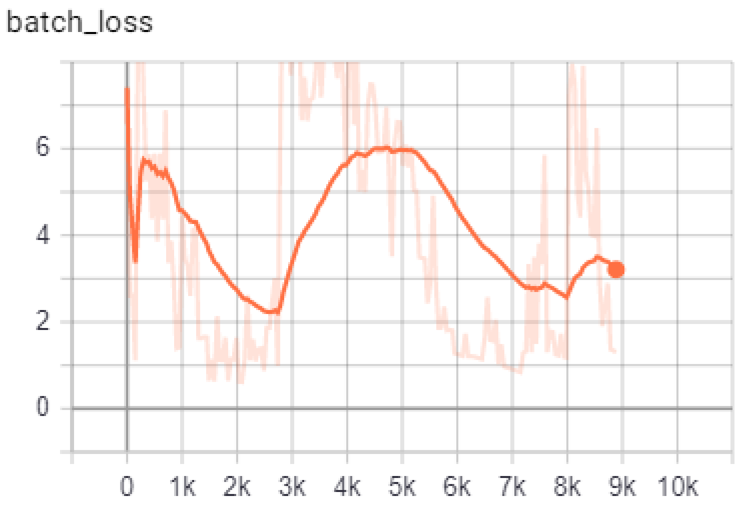


Fig.20 SVM Batch Loss vs Steps (Set-1)

SVM gave the test accuracy peak of **98.95%**

5.2.1.4 Logistic Regression

Figures 21 and 22 show the accuracy and loss graphs for Logistic Regression (LR) after training on Set-1.

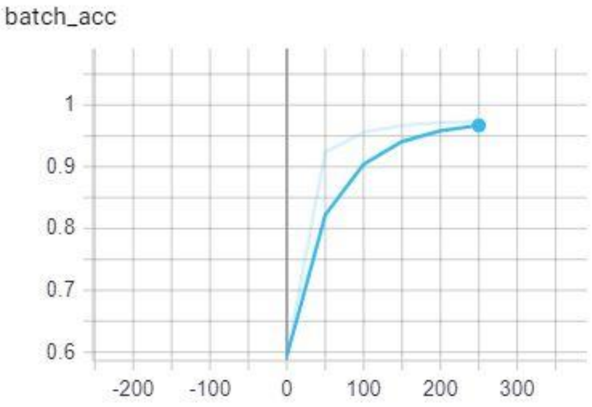


Fig.21 LR Batch Accuracy vs Steps (Set-1)

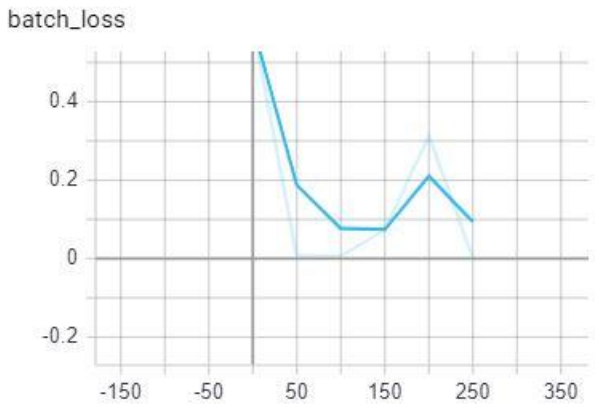


Fig.22 LR Batch Loss vs Steps (Set-1)

Logistic Regression gave the test accuracy peak of **99.85%**

Table 4 shows the test accuracies of the models trained on Set-1.

Table 4: Model Accuracies on Set-1

Model	**Test Accuracy (%)**
CNN	100
ResNet	99.94
SVM	98.95
LR	99.85

5.2.2 Set-2 Results

In this set, the classes consist of Real Face Images and the images generated by Reface App [10] which implicitly uses a variety of GANs and DeepFaceLab[11]. The models trained are CNN, ResNet architecture, SVM and Logistic Regression.

5.2.2.1 CNN

Figures 23 and 24 show the accuracy and loss graphs for CNN after training on Set-2.

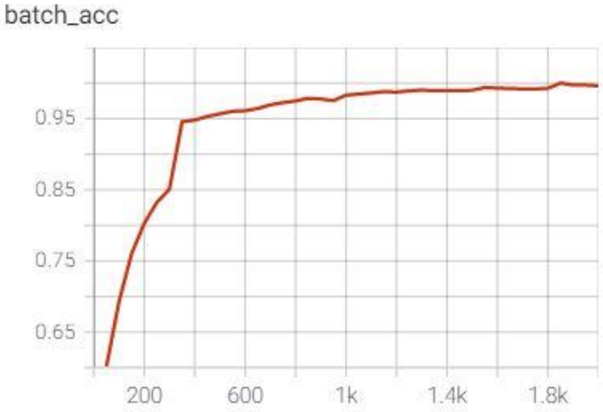


Fig.23 CNN Batch Accuracy vs Steps (Set-2)

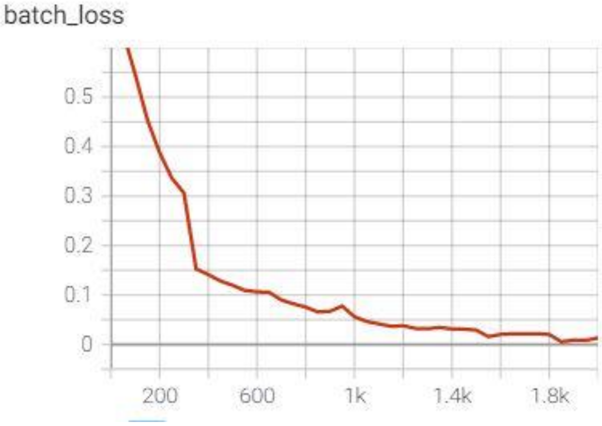


Fig.24 CNN Batch Loss vs Steps (Set-2)

CNN gave the test accuracy peak of **95.61%**

5.2.2.2 ResNet

Figures 25 and 26 show the accuracy and loss graphs for ResNet after training on Set-2.

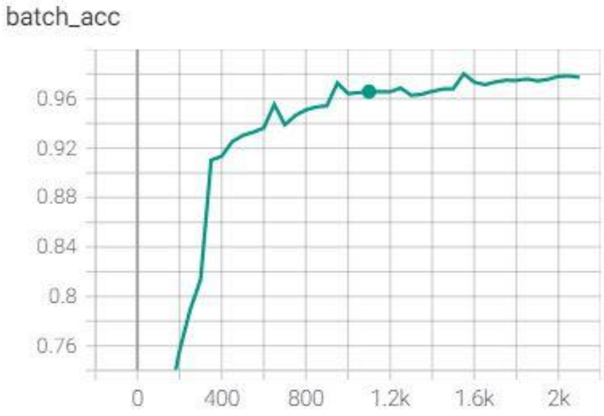


Fig.25 ResNet Batch Accuracy vs Steps (Set-2)

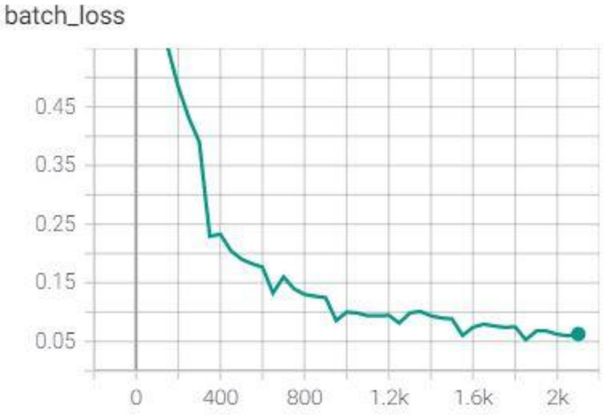


Fig.26 ResNet Batch Loss vs Steps (Set-2)

ResNet gave the test accuracy peak of **92.97%.**

5.2.2.3 SVM

Figures 27 and 28 show the accuracy and loss graphs for SVM after training on Set-2.

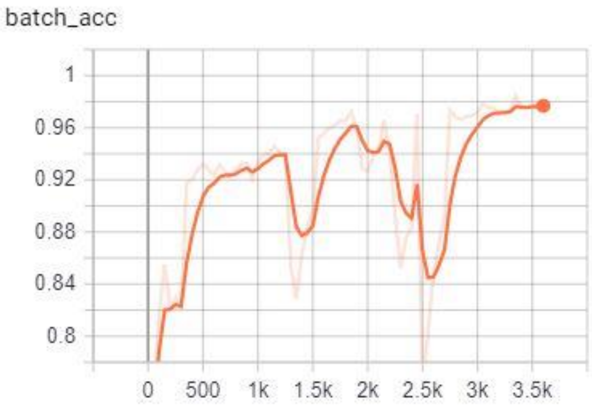


Fig.27 SVM Batch Accuracy vs Steps (Set-2)

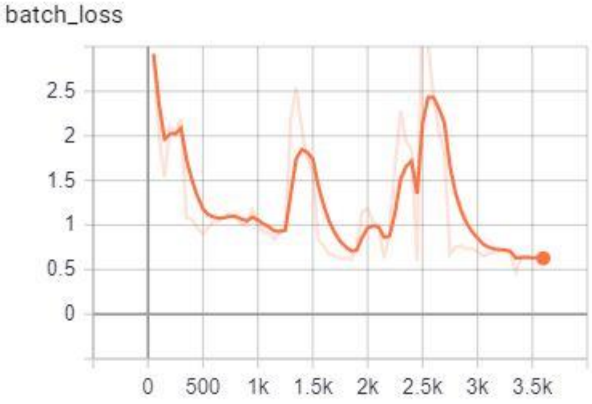


Fig.28 SVM Batch Loss vs Steps (Set-2)

SVM gave the test accuracy peak of **95.13%**

5.2.2.4 Logistic Regression

Figures 29 and 30 show the accuracy and loss graphs for Logistic Regression (LR) after training on Set-2.

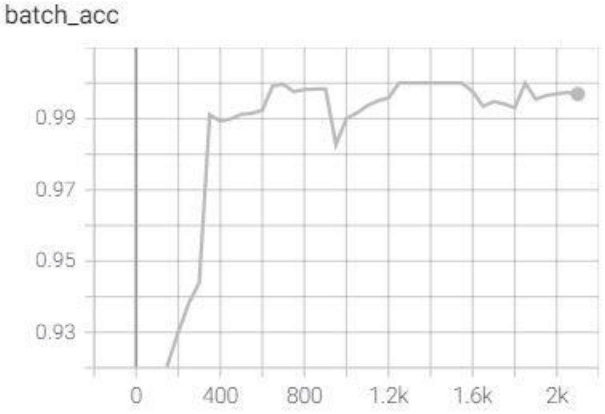


Fig.29 LR Batch Accuracy vs Steps (Set-2)

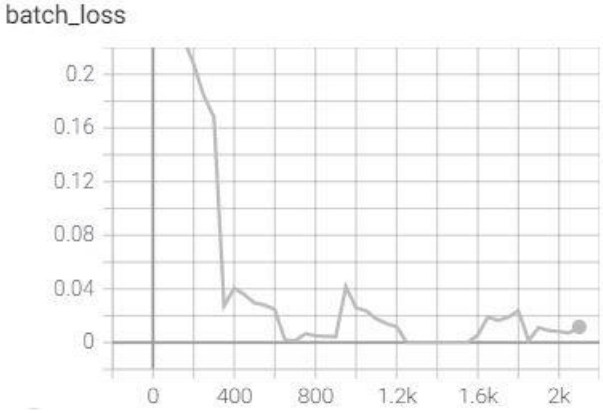


Fig.30 LR Batch Loss vs Steps (Set-2)

Logistic Regression gave the test accuracy peak of **97.96%**

Table 5 shows the test accuracies of the models trained on Set-2.

Table 5: Model Accuracies on Set-2

Model	**Test Accuracy (%)**
CNN	95.61
ResNet	92.97
SVM	95.13
LR	97.96

In the second set, Logistic Regression model performed the best with **97.96%** accuracy

From the results shown in Set-1 and Set-2, it has been observed that the models performed slightly better with Set-1 than Set-2. This is because the fake images in Set-1 are only generated by StyleGAN, in contrast to the ones generated by a variety of GANs and DeepFaceLab in the case of Set-2 using Reface App. It gives the conclusion that the fake face images generated by StyleGAN are weaker and easier to distinguish from the real face images, than the images generated by Reface app. This can be due to the possibility of better upsampling strategies incorporated by Reface App as explained in [24], and also a stronger generator and discriminator architecture used. However, the models perform really well with high accuracy scores with both the sets. This proves that the frequency spectrum analysis, using DCT, is an efficient way to counter GAN generated images as it finds out some distinguishing features in the fake images. Hence, this technique can be implemented to detect presentation attacks using deepfakes.

6. Conclusion and Future Scope

In this project, we have targeted on two aspects of face liveness detection or presentation attack: presentation attacks using photos or 2D paper mask, and deepfake attacks. For photo attacks, we observed that interactive action like eye blinking works effectively against static photo attacks. The model gave high accuracy on diverse dataset of eyes, and OpenCV Haar-Cascade classifier and HOG worked effectively for real time face detection and face-encoding comparison. For future work, the model could be trained for dynamic attacks like video attacks and 3-D rendered face masks. One approach could be to emphasize on the texture and image detail, and also on the reflection properties. This is because a rendered mask or video might consist of noise and blurs, and also light would reflect differently on a live face than a video. Another approach for tackling video attacks could be to employ depth estimation in real time on webcam video frames. This is because a 3-D live face would portray depth more, than a video on a 2-D screen.

For the deepfake detection, we extensively analyzed the frequency spectrum of the images generated from StyleGAN and different GAN combinations. It was observed that images generated by GAN models possess some common features in the frequency spectrum which is different from natural images. It is probable that these features and vulnerabilities are due to the upsampling techniques or operations incorporated by GAN architectures. Hence, log of Type-2 DCT coefficients are employed. It was also observed that simple linear models like SVM and Logistic Regression, and elaborate CNN-based models like ResNet perform with great accuracies in the frequency domain, proving the approach effective. This supports the claims and verifies the results shown in [24] for different datasets as well. For future work, we can work according to advancing GAN architectures which incorporate upsampling operations efficiently without leaving any vulnerabilities or features making them separable from live faces. Also, another approach can be to handle real-time deepfake generator on webcam, which makes analyzing frequency spectrum of images in real time overwhelming.

REFERENCES

[1] Thales, “Liveness in biometrics: spoofing attacks and detection,” https://www.thalesgroup.com/en/markets/digital-identity-and-security/government /inspired /liveness-detection, Dec. 2020

[2] G. Kim, S.Eum, J. K. Suhr, D. I. Kim, K. R. Park, and J, Kim, “Face liveness detection based on texture and frequency analyses,” *5th IAPR Intl Conf on Biometrics (ICB),* New Delhi, India, pp. 67-72, Mar. 2012

[3] M. Somers, “Deepfakes, explained,” https://mitsloan.mit.edu/ideas-made-to-matter/deepfakes-explained, Jul. 2020

[4] P. Korshunov and S. Marcel, “Vulnerability assessment and detection of Deepfake videos,” Idiap Research Institute, Martigny, Switzerland, 2019

[5] V. R. Gosavi, G. Sable and A. Deshmane, “Evaluation of Feature Extraction Techniques using Neural Network as a Classifier: A Comparative Review for Face Recognition” *IJSRST 2018,* vol. 4, Issue 2, 2018

[6] C. Li, Y. Yang, M. Feng, S. Chakradhar and H. Zhou, “Optimizing Memory Efficiency for Deep Convolutional Neural Networks on GPUs,” *Proc of the Int. Conf for High Perf Comp, Netw, Strg and Anls*, pp. 633-644, 2016

[7] R. Thaware, “Real-Time Face Detection and Recognition with SVM and HOG Features,” https://www.eeweb.com/real-time-face-detection-and-recognition-with-svm-and-hog-features/#:~:text=A%20HOG%20is%20a%20feature%20descriptor%20generally%20used%20for%20object%20detection.&text=In%20the%20current%20example%2C%20all,per%20pixel%20of%20the%20image, May 2018

[8] S. Chang, D. Xiaoqing and F. Chi, “Histogram of the Oriented Gradient for Face Recognition,” TSINGHUA Science and Technology, ISSN 1007-0214, 15/15, pp. 216-224, vol. 16, number 2, Apr. 2011

[9] S. Chakraborty and D. Das, “An Overview of Face Liveness Detection,” *Intl J on Info Thry (IJIT)*, vol. 3, no. 2, Apr. 2014 Thales, “Liveness in biometrics: spoofing attacks and detection,” https://www.thalesgroup.com/en/markets/digital-identity-and-security/government /inspired /liveness-detection, Dec. 2020

[10] N. Thomas, “Deepfake video app Reface is just getting started on shapeshifting selfie culture,” https://techcrunch.com/2020/08/17/deepfake-video-app-reface-is-just-getting-started-on-shapeshifting-selfie-culture/, Aug. 2020

[11] I. Perov et al., “DeepFaceLab: A simple, flexible and extensible face swapping framework,” *arXiv*: 2005.05535v4, May 2020

[12] Starlink, “Biometrics Face Recognition - How does it Work?,” https://www.starlinkindia.com/blog/biometrics-face-recognition/, May 2019

[13] S. Saha, “A Comprehensive Guide to Convolutional Neural Networks – the ELI5 way,” https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53, Dec. 2018

[14] Tencent, “Biometric Authentication Under Threat: Liveness Detection Hacking,” https://docplayer.net/151329328-Biometric-authentication-under-threat-liveness-detection-hacking.html, blackhat USA 2019

[15] S. Kim, S.Yu, K. Kim, Y. Ban and S. Lee, “Face liveness detection using variable focusing,” *Intl Conf 2013*, Biometrics (ICB), pp 1-6, 2013

[16] R. Tolosana, R. V. Rodriguez, J. Fierrez, A. Morales and J. O. Garcia, “DeepFakes and Beyond: A Survey of Face Manipulation and Fake Detection,” *arXiv*:2001.00179v3, Jun. 2020

[17] “Deepfake” https://en.wikipedia.org/wiki/Deepfake, Accessed on: (05/01/2021)

[18] J. Brownlee, “A Gentle Introduction to Generative Adversarial Networks (GANs),” https://machinelearningmastery.com/what-are-generative-adversarial-networks-gans/, 2019

[19] J. Brownlee, “18 Impressive Applications of Generative Adversarial Networks (GANs),” https://machinelearningmastery.com/impressive-applications-of-generative-adversarial-networks/, 2019

[20] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. warde-Farley, S. Ozair, A. Courville and Y. Bengio, “Generative adversarial networks,” 2014

[21] M. Arjovsky and L. Bottou, “Towards principled methods for training generative adversarial networks,” 2017

[22] T. Karras, S. Laine and T. Aila, “A Style-Based Generator Architecture for Generative Adversarial Networks,” *arXiv*:1812:04948v3, Mar. 2019

[23] “Which Face Is Real?,” https://www.whichfaceisreal.com/, Accessed on: (03/06/2021)

[24] J. Frank, T. Eisenhofer, L. Schonherr, A. Fischer, D. Kolossa and T.Holz, “Leveraging Frequency Analysis for Deep Fake Image Recognition,” *arXiv*: 2003.08685v3, Jun. 2020

[25] “Face Recognition,” https://face-recognition.readthedocs.io/en/latest/readme.html, Accessed on: (10/12/2020)

[26] “Dlib C++ Library,” http://dlib.net/, Accessed on: (10/12/2020)

[27] “Face Detection using Haar Cascades,” https://docs.opencv.org/3.4.3/d7/d8b/tutorial_py_face_detection.html, Accessed on: (10/14/2020)

[28] "Closed Eyes In the Wild (CEW)," http://parnec.nuaa.edu.cn/_upload/tpl/02/db/731/template731/pages/xtan/ClosedEyeDatabases.html, Accessed on: (09/15/2020)

[29] G. B. Huang, M. Ramesh, T. Berg and E. Learned-Miller, "Labeled faces in the wild: A database for studying face recognition in unconstructed environments," *Tech Rep* 07-49, University of Massachusetts, Amherst, Oct. 2007

[30] "Flickr-Faces-HQ Dataset (FFHQ)," https://github.com/NVlabs/ffhq-dataset, Accessed on: (03/17/2021)

[31] "Legacy discrete Fourier transforms (scipy.fftpack)," https://docs.scipy.org/doc/scipy/reference/fftpack.html, Accessed on: (03/17/2021)